Virtualised e-Learning with Real-Time Guarantees on the IRMOS Platform[1]

T. Cucinotta, F. Checconi
Scuola Superiore Sant'Anna
Pisa, Italy
{t.cucinotta,f.checconi}@sssup.it

G. Kousiouris, D. Kyriazis, T. Varvarigou
National Technical University of Athens
Athens, Greece
{gkousiou,dkyr,dora}@mail.ntua.gr

A. Mazzetti
GIUNTI Labs
Sestri Levante, Italy
a.mazzetti@giuntilabs.com

Z. Zlatev, J. Papay, M. Boniface
University of Southampton IT Innovation Centre, Southampton, UK
{zdz,jp,mjb}@it-innovation.soton.ac.uk

S. Berger, D. Lamp
University of Stuttgart
Stuttgart, Germany
{soren.berger,dominik.lamp}@rus.uni-stuttgart.de

T. Voith, M. Stein
Alcatel Lucent
Stuttgart, Germany
{thomas.voith,manuel.stein}@alcatel-lucent.de

Abstract—**In this paper we focus on how Quality of Service guarantees are provided to virtualised applications in the Cloud Computing infrastructure that is being developed in the context of the IRMOS[1] European Project. Provisioning of proper timeliness guarantees to distributed real-time applications involves the careful use of real-time scheduling mechanisms at the virtual-machine hypervisor level, of QoS-aware networking protocols and of proper design methodologies and tools for stochastic modelling of the application. The paper focuses on how we applied these techniques to a case-study involving a real e-Learning mobile content delivery application that has been integrated into the IRMOS platform and its achieved performance.**

Keywords – Real-time scheduling; virtualised infrastructures; stochastic modelling; e-Learning.

I. Introduction

Nowadays, with the advent of affordable high-speed Internet connections, distributed computing is becoming a predominant software development paradigm. Applications are developed for, and made available within, distributed infrastructures, where users can easily access remote services and exploit remote resources from their local workstation, laptop, palmtop device and/or mobile phone. In the Cloud Computing model, distributed applications are developed by Software-as-a-Service (SaaS) providers, by means of tools made available by Platform-as-a-Service (PaaS) providers, for being deployed over the resources made available by Infrastructure-as-a-Service (IaaS) providers. The viability of IaaS is dependent upon the use of virtualisation technologies. Virtualised computing allows for deploying multiple virtual machines (VMs), hosting multiple Operating Systems and services therein, over the same physical hosts, thus achieving a better server consolidation level. Also, thanks to network virtualisation techniques, it is possible to migrate the VMs from a physical host to another one, for maintenance or load-balancing reasons, in a seamless manner.

In this evolving scenario, more and more distributed applications with tight interactivity and timing requirements are being deployed over virtualised IaaS infrastructures. However, when multiple VMs are deployed over the same physical resources (e.g., links and CPUs), the level of performance experienced by each VM is not stable any more, but it depends heavily on the overall workload imposed by the other VMs competing for the shared resources. However, using proper scheduling technologies, coupled with proper performance modelling techniques, it is possible to deploy virtualised distributed applications with a stable performance level, as being experimented with the virtualised Cloud Computing infrastructure for service-based computing that is being developed in the context of the IRMOS[2] European Project. In this paper, we show how these concepts have been practically applied to a real e-Learning application.

II. Related Work

In this section, related works appeared in the research literature are briefly summarised.

The work by Lin and Dinda in [4] presents various similarities with the one presented in this paper. First, an EDF-based scheduling algorithm for Linux is used on the host to schedule Virtual Machines (VMs). Furthermore, an analysis is conducted on the application performance, investigating the effects of scheduling decisions and concurrent virtual machines execution. The analysis is very thorough and interesting, however the major limitation of the work resides in the way low-level scheduling is achieved. In fact, the authors make use of a scheduler built into a proper user-space process (VSched), which exploits POSIX real-time priorities in order to achieve an EDF-based scheduling of VMs, and SIGSTOP/SIGCONT signals for realising optionally hard resource reservations. Such an approach presents high scheduling overheads due to the forcibly increased number of context switches, whilst our scheduler [7] is directly built into the kernel and does not introduce any additional context switch; also, VSched cannot properly react to those situations in which a VM blocks or unblocks, e.g., as due to I/O operations, something that is needed in order to guarantee a proper level of temporal isolation, like done instead in our scheduler by exploiting the CBS algorithm [8]. Another very interesting work by the same authors is [6], where the users of a virtual machine are given the opportunity to adapt the

1 The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7 under grant agreement n. 214777 "IRMOS—Interactive Realtime Multimedia Applications on Service Oriented Infrastructures".

2 More information at: http://www.irmosproject.eu/.

allocated CPU through a simple interface, based on their experience with the application. The cost of the increase is shown, so that the user may decide on the fly. While it is a very promising approach and would eliminate a vast number of issues with regard to application QoS levels, its main drawback is in cases of workflows. Inside a workflow, a degradation in performance may be due to a bottleneck on various nodes executing a part of it. The user will most likely be unaware of the location of the bottleneck, especially in cases of non experts. Instead, the work by Nathuji et al. [17] focuses on automatic on-line adaptation of the CPU allocation in order to keep a stable performance of VMs. However, the framework cannot see a VM as a "black-box", but it needs an application-specific metrics in order to run the necessary QoS control loop, going beyond the common IaaS business model.

Gupta et al. investigated on the performance isolation of virtual machines [16], focusing on the exploitation of various scheduling policies available in the Xen hypervisor [18]. However, in this work we focus on the KVM[3] hypervisor, along the lines of other works of ours in which we showed how to provide isolation of compute-intensive [19] and network-intensive [20] VMs. Instead, here we also address the modelling issues related to the deployment of an e-Learning application with proper QoS guarantees.

Shirazi et al. [3] proposed DynBench, a benchmark for infrastructures supporting distributed real time applications. This creates dynamic conditions for the testing of the infrastructures. While promising, this framework is mainly oriented towards investigating the limits of the infrastructure and not towards understanding the application behaviour in relationship to different scheduler configurations.

In [5], Germain et al. present DIANE for Grid-based user level scheduling. However, the focus is on controlling the execution end time of long processing applications, and not on real time interactive ones as done in this paper.

In terms of application performance modelling in distributed infrastructures a number of interesting works exist. A code analysing process that allows for the simulation of system performance is described in [11]. It models the application by a parameter-driven Conditional Data Flow Graph (CDFG) and the hardware (HW) architecture by a configurable HW graph. The execution cost of each task block in the application CDFG is modelled by user-configurable parameters, which allows for highly flexible performance estimation. The simulator takes the application CDFG and HW graph as the input and performs a low-level simulation to catch the detailed HW activities. While very promising, it needs the source code in order to provide the CDFG. In our work, we deal with VMs as black boxes, what allows for the deployment of applications where the source code is not available for confidentiality purposes.

Another interesting work is presented by Lee et al. in [12]. The application, whose performance must be measured, is run under a strict reservation of resources in order to determine if the given set of reservation parameters satisfies the time constraints for execution. If this is not the case, then these parameters are altered accordingly. If there is a positive surplus, the resources are decreased and if it is negative they are increased until a satisfying security margin is reached. While assuring high utilisation rates, the main disadvantage of this methodology is that this must be performed for every individual execution with the specific SLA parameters before the actual deployment.

Bekner et al. introduce the Vienna Grid Environment (VGE) [13], a framework for incorporating QoS in Grid applications. It uses a performance model to estimate the response time and a pricing model for determining the price of a job execution. In order to determine whether the client's QoS constraints can be fulfilled, for each QoS parameter a corresponding model has to be in place. However, VGE does not prescribe the actual nature of performance models. It specifies only an abstract interface for performance models, taking as granted that these models will be provided from analytical modelling or historical data. But analytical modelling in general requires a thorough knowledge of the application, in order to deduct the equations that depict its performance. In this work we also use analytical models, however they are coupled with a black box approach for the parts of the application that are not visible to the external world besides the developer. Other works exist that address QoS assurance in GRID environments focusing on performance prediction [25] and control via service selection [26].

While numerous promising solutions exist to the problem of performance analysis of VMs in presence of real-time scheduling, these either are not focused on critical parameters that are necessary for running real time applications on SOIs, or they lack for a proper low-level real-time scheduling infrastructure which is needed for supporting temporal isolation among concurrently running VMs.

III. Performance Isolation – The IRMOS/ISONI Way

One of the core components which is being developed in IRMOS is the Intelligent (virtualised) Service-Oriented Networking Infrastructure (ISONI) [15]. It acts as a Cloud Computing[4] IaaS provider for the IRMOS framework and manages a set of physical computing, networking and storage resources available in form of multiple nodes/sites within a provider domain (see Figure 1).

ISONI provides those visualised resources over which IRMOS applications are deployed. One of the key innovations introduced by ISONI is its capability to ensure guaranteed levels of resource allocation for each individual application instance hosted within the ISONI domain.

This is realised by allowing applications to be deployed in form of a Virtual Service Network (VSN). This is a graph whose vertices represent individual Service Components (SCs) of an application which may be deployed in form of

3 More information at: http://www.linux-kvm.org.

4 More information at: http://www.cloudcomputing.org/.

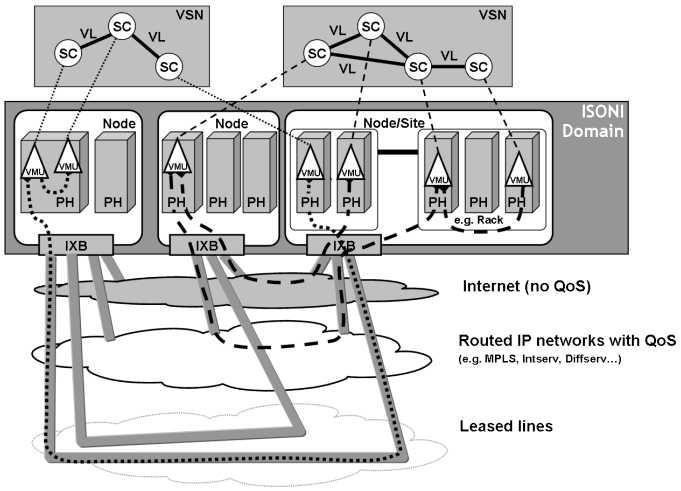

Figure 1: Deployment of Service Components (SCs) within Virtual Machine Units (VMUs) over IRMOS/ISONI.

Virtual Machine Units (VMUs), and whose edges represent communications – the virtual links (VLs) – among them.

In order for the system represented by a VSN to comply with real-time constraints as a whole, QoS needs to be supported for all the involved resources, particularly for network links, computing elements (CPUs) and storage resources. To this purpose, VSN elements are associated with precise resource requirements, e.g., in terms of the required computing power for each node and the required networking performance (i.e., bandwidth, latency, jitter, etc.) for each link. These requirements are fulfilled thanks to the allocation and admission control logic pursued by ISONI for instantiating VMs within the managed set of available physical resources, and to the low-level mechanisms shortly described in what follows (a comprehensive ISONI overview is out of the scope of this paper and can be found in [15]).

A. Isolation of Computing

In order to provide scheduling guarantees to individual VMs scheduled on the same system, processor and core, IRMOS incorporates a hybrid deadline/priority (HDP) real-time scheduler [7] developed within the IRMOS consortium for the Linux kernel. This scheduler provides temporal isolation among multiple possibly complex software components, such as entire VMs (with the KVM hypervisor[3], a VM is seen as a process). It uses a variation of the Constant Bandwidth Server (CBS) algorithm [8], based on Earliest Deadline First (EDF), for ensuring that each group of processes/threads is scheduled for Q time units (the budget) every interval of P time units (the period). The CBS algorithm has been extended for supporting multi-core (and multi-processor) platforms, achieving a partitioned scheduler where the set of tasks belonging to each group may migrate across the associated CBS scheduler instances running on different CPUs, according to the usual multi-processor real-time priority-based scheduling in Linux.

The scheduler exhibits an interface towards user-space applications based on the cgroups [14] framework, which allows for configuration of kernel-level parameters by means of a filesystem-based interface. This interface has been wrapped within a Python API, in order to make the real-time scheduling services accessible from within the IRMOS platform. The parameters that are exposed by the scheduler are the budget Q and the period P, as explained above.

B. Isolation of Networking

Isolation of the traffic of independent VMs within ISONI is achieved by a VSN-individual virtual address space and by policing the network traffic of each deployed VSN. The two-layer address approach avoids unwanted crosstalk between services sharing physical network links. The traffic policing avoids that the network traffic going through the same network elements causes any overload leading to an unduly uncontrolled growth of loss rate, delay and jitter for the network connections of other VSNs. A gap-less policing ensures that the network multiplex stages always get a controlled load of traffic. Therefore, bandwidth policing is an essential building block to ensure QoS for the individual virtual links. It is important to highlight that ISONI allows for the specification of the networking requirements in terms of common and technology-neutral traffic characterisation parameters, such as the needed guaranteed average and peak bandwidth, latency and jitter. An ISONI transport network adaptation layer abstracts from transport network technology-individual QoS mechanism like Diffserv [21], Intserv [22][23] and MPLS [24]. Depending on the specified networking requirements, an adequate transport network is chosen in order to meet the applications requirements, which could be transport networks without any QoS like the Internet, transport networks with QoS mechanisms with or without reservations or leased lines (see Figure 1).

Then, the networking protocols and technologies which are most appropriate for the underlying hardware are used in order to meet the applications requirements.

C. Modelling and Benchmarking

One of the key steps in deploying applications with precise real-time or generally QoS guarantees within IRMOS is the one of building a performance model of the application behavior. This means that, given application-specific configurable parameters (e.g., number of users, resolution of multimedia contents, etc.), and given possible performance levels that one may want to achieve, it should be possible to determine what allocation is needed on the physical resources in order to accomplish that. This is a core information needed by the SaaS provider in order to establish an accurate pricing policy for the customer(s).

Application performance in the cloud depends on many complex factors such as application workload, conditions of the network paths between the user(s) and the server(s) and the computing workload of the physical host(s). Computing workload factors are especially significant in multi-tenant clouds where single hosts are used to service multiple applications. However, the ISONI support for temporal isolation of VMs with guaranteed QoS means the interference due to shared resources becomes negligible. The immediate advantage of this is that the performance of an individual application to be instantiated within the platform may be easily benchmarked and modelled as a pure function of its

application-specific parameters and the amount of allocated resources, and it turns out to be independent of whatever else is (and/or will be) instantiated in the domain by the provider.

IV. The e-Learning Application

We focus on an e-Learning mobile instant content delivery application, developed taking advantage of a service-oriented architecture paradigm, in which real-time requirements play an important role. In this scenario a user can receive on his/her mobile phone some e-Learning contents relevant to the position where he/she is (e.g., walking near to historical monuments). It consists of a Tomcat based e-Learning server that exploits a MySQL database for content management (see Figure 3). The application is able to receive queries with GPS data from multiple clients, search the database and respond with e-Learning contents corresponding to the provided GPS coordinates (see Figure 2). The application server is provided as a Web Application Archive (war) file, installed in Tomcat, and made available as a Virtual Machine image within the IRMOS infrastructure. Using ISONI, each instance of the application can be assigned precise computing and networking resources that ensure the high-level requirements defined within an application provider's Service-Level Agreement (SLA) can be met with an agreed level of reliability.

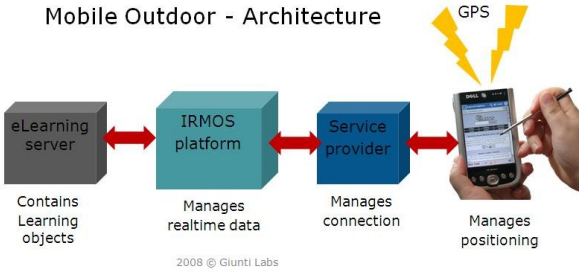

Figure 2. Components of the e-Learning application.

The timing requirements of the application are mainly related to the response times of the individual requests submitted by the multitude of users. As discussed in Section III.C., thanks to the deployment within IRMOS, these response times depend merely on application-specific parameters, i.e., on the number of concurrent users querying the same e-Learning instance and the size of the downloaded contents.

A. Application Client Simulation Description

In order to investigate application performance, we developed a multi-user client simulator. This is capable of simulating the random movements of a certain number of users walking around given GPS coordinates. Then, the simulator mimics the behaviour of the real mobile client associated with the application: whenever the monitored GPS coordinates move sufficiently away from the position of the last queried content, a new request is submitted to the server with the new user position. The number of users and a few parameters governing how each emulated user exactly moves (e.g., the user speed) determine the exact pattern of requests submitted to the server, thus strongly impacting on the imposed server load.

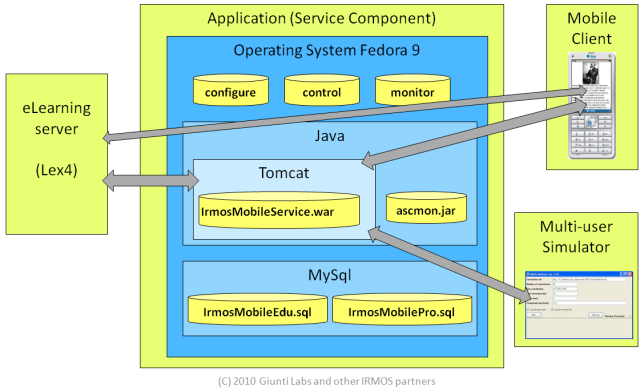

Figure 3: Software architecture of the e-Learning application.

V. Performance Estimation

In order to estimate what QoS level can be achieved using different resource configurations, we use performance modelling techniques. Many times, the application internal software structure may be too complex to be modelled. Or, it may be unknown because developers are reluctant to share detailed information about their application internals, for confidentiality purposes. In other cases, the use of external libraries or components whose internals are unknown makes it

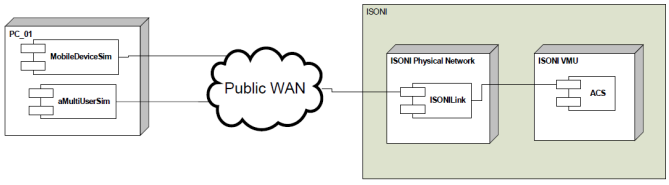

Figure 4: Modelled elements of the e-Learning application.

impossible to build an exact model. So, from a modelling point of view, it is critical to be able to identify the expected QoS output using a *black-box* approach.

Therefore, we use a combination of a stochastic model for predicting statistics over the expected run-time networking performance, and an Artificial Neural Network (ANN) for identifying the dependency of a component QoS from factors

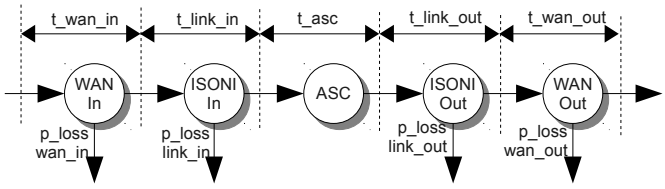

Figure 5: Stochastic performance model: t_wan_in and t_wan_out are modelled as exponential distributions, the other delays as Erlang ones.

like application-level parameters (e.g., number of clients) or scheduling parameters (e.g., allocated budget and period). These two models, put together, allow for a precise estimate of the overall end-to-end QoS experienced by end-users.

A. Stochastic Performance Model

We built a Matlab model for simulating, by means of Monte-Carlo type discrete event simulation, a system composed of

(see Figure 4): a request generator (modelling the end users), a Public Wide Area Network (WAN), a Private Network internal to an ISONI domain and the VMU hosting the actual Application Service Component (ASC). In order to account for interactivity, we modelled both paths from the user to the ASC and the other way round. The model uses a mix of exponential and Erlang probability distributions (see Figure 5) for modelling the latencies of application requests while traversing the involved networks, and it may also simulate packet loss due to buffer saturation in the various networks (particularly useful for UDP-based communications).

The individual parameters of the model need to be tuned by resorting to proper benchmarking techniques. The behaviour of the latencies inside the ISONI internal network may be accurately estimated thanks to the ISONI networking isolation, and they depend merely on the requested network-level QoS parameters specified in the VSN, and the requests pattern. On the other hand, parameters relative to the QoS-unaware WAN must be estimated based on available statistics on the overall network workload foreseen at the time of usage of the application. However, a widespread usage of ISONI would reduce the need for traversing QoS-unaware networks.

The behaviour of the ASC was also estimated as an Erlang distribution. However, due to the non-trivial dependency of the performance from application-level parameters, in addition to the resource allocation ones, the Erlang parameters were tuned by resorting to an ANN model (see below).

The described simulator is capable of providing, for each configuration, the full probability distribution of the end-to-end response-times, as well as simpler statistics that may be easily leveraged at design-time, such as the average or a given percentile of the distribution. For example, this allows for finding the configuration parameters granting a given end-to-end response-time with a given probability.

B. Artificial Neural Networks in the Model

An ANN model is used for modelling the time the server needs in order to retrieve the results from the internal database. The factors that are taken under consideration are mainly the number of connected clients and the scheduling decisions (Q and P parameters). Following the black-box approach, the use of ANNs allows for an easy addition of further inputs (or outputs) as needed (e.g., hardware-specific parameters like the reserved memory or processor speed), once the necessary training data sets are collected.

The investigation of the effect of parameters like the allocated CPU time Q over a period P is critical due to its influence in the QoS output. The choice of P is mainly driven by the time granularity for the allocation needed by the application. For example, for interactive applications, with fast response times and relatively light computations, the granularity must be kept small (in the order of 10–100 ms). For scientific applications performing long and heavy computations, large periods will result in lower overheads (500 ms and beyond).

The outputs of the ANN have been chosen to represent the average and the standard deviation of the expected response times, as due to the configuration represented at the ANN inputs. These outputs are easily mapped to the Erlang parameters needed for modelling the ASC temporal behaviour in the general model in Figure 5. The ANN model structure is described in V.C., while the data gathered as training set is presented in VI.B.

C. ANN Structure and Design

In order to implement the ANN, a standard form of network was selected. The type of operation that was desired was function approximation, in order to determine the effect of the input parameters (number of clients, Q, P) on the predicted output (mean value of inner server response time and standard deviation). The collection of the data set was performed with the process described in [1]. Two more inputs were included, CPU speed and VM memory size, but the main focus was on the initial 3 parameters. The resulting network for the mean time prediction was a 3-layer, feed-forward, back propagation network, created through the GNU Octave tool[5]. It was trained with the Octave `trainlm` function, using the Levenberg-Marquardt algorithm [2]. The structure of the network is shown in Table 1. All the inputs and outputs are normalized in the (-1,1) interval. A standard form of function approximation network was used, with one hidden layer and Tansig transfer functions for the input and hidden layer and linear transfer function for the output layer.

For the standard deviation, a similar process was followed (but with the Logsig transfer function) and the resulting network appears in Table 2.

Layer	Transfer Function	Size (Neurons)	Mean Absolute Error
Input Layer	Tansig	5	2.51%
Hidden Layer	Tansig	2	
Output layer	Linear (Purelin)	1	

Table 1: Structure of Mean Response Time Prediction ANN.

Layer	Transfer Function	Size (Neurons)	Mean Absolute Error
Input Layer	Logsig	5	2.75%
Hidden Layer	Logsig	2	
Output layer	Linear (Purelin)	1	

Table 2: Structure of Standard Deviation Prediction ANN.

VI. Experimental Results

In this section we report experimental results validating the presented approach to the provisioning of performance guarantees to the e-Learning application by means of proper real-time scheduling and modelling techniques.

First, the assumptions of temporal isolation over which the modelling technique relies are validated. Then, some

5 More information is available at: http://www.gnu.org/software/octave/.

experimental data used for training the ANN models is described, and finally the accuracy of the ANN-based estimations is discussed.

A. Temporal Isolation by Real-Time Scheduling

We ran an experiment for the purpose of validating our approach to the temporal isolation of VMs concurrently running on the same CPU based on real-time scheduling. To this purpose, we considered two instances of the e-Learning application deployed on the same host and physical core. We launched two instances of the e-Learning multi-user simulator submitting requests coming from 10 emulated users to the two servers, from a second machine in an isolated networking context. We collected the response-times experienced by the two multi-user simulator instances under various conditions in terms of the scheduling parameters configured for the two VMs on the server host.

In Figure 6 we report the average response-time of the first VM as a function of the CPU share (on the x-axis) assigned to

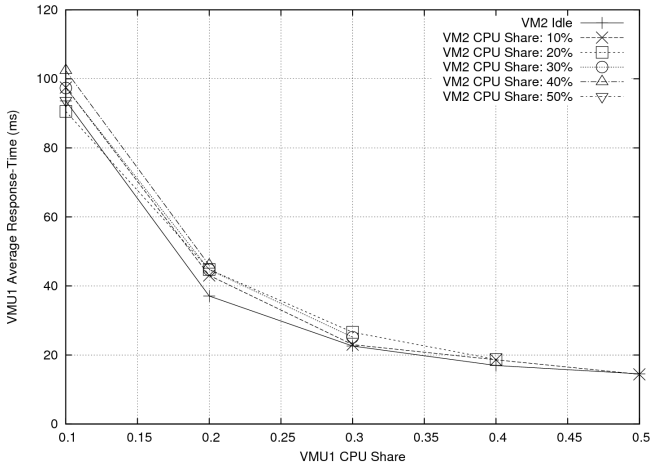

Figure 6: Average response-time of the first VM as a function of the CPU share (on the x-axis) assigned to it, at varying CPU shares assigned to the second competing VM (corresponding to the various curves).

it, at varying CPU shares assigned to the second competing VM (corresponding to the various curves). Under the ideal conditions of perfect temporal isolation, we would like the second competing VM workload, ranging from completely idle (continuous curve) to having a 50% of load on the system (dashed curve tagged with little triangles), to have no impact at all on the performance of the first VM. This would correspond to having all the curves perfectly superimposed. As it can be seen from the experimental results, the real-time scheduler achieves a nearly good approximation of such a condition, realising a set of curves which are quite close to each other, where the increase of computing resources granted to the second VM corresponds to a slight decrease of the performance of the first VM. This may be mainly attributed to an increased contention on the cache, and constitutes a minimum of interference which cannot be removed. Other factors of interference which are not trivial to keep under control are due to shared resources on the host OS, like the networking stack and interrupts. For example, see [20] for a

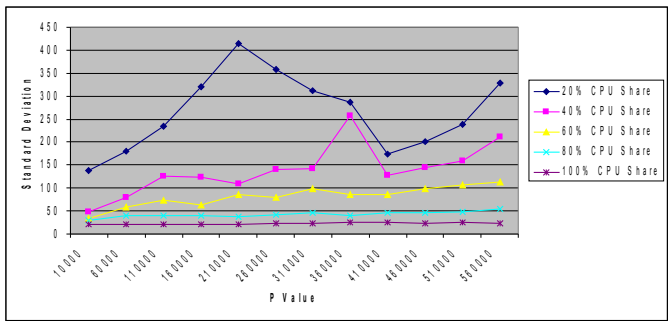

Figure 7: Standard Deviation with regard to changing P for 90 users.

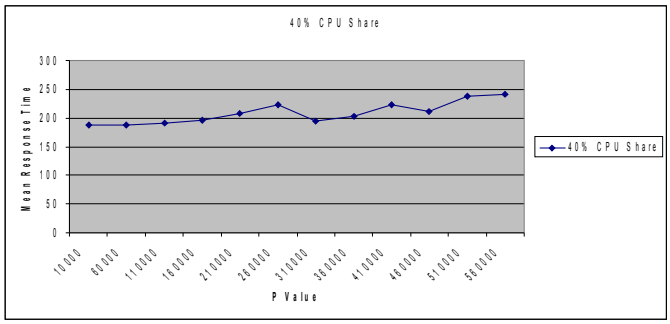

Figure 8: Mean value with regard to changing P for 70 users and 40% CPU share.

discussion of the interference due to network-intensive VMs, and the extent to which it can be controlled by real-time scheduling of the CPU.

B. Experimental Performance of the e-Learning Application

In this section, experimental performance data gathered for various configurations of both application-level and resource allocation parameters is presented. The range of values that were altered for the configuration parameters are:

- Number of Users: 30-150
- Q/P (CPU share) : 20-100% with a step of 20
- P: 10000–560000 (μsec) with a step of 50000

For each configuration, about 800 response times were collected, and the corresponding average and standard deviation figures computed. An indicative set of these measurements is discussed below.

The effect of changing granularity on the deviation of the response time values can be observed in Figure 7. This is expected since with high values of P, the service has long active and inactive periods. If the requests fall in the active interval, they will be satisfied quickly but if they fall in the inactive one then they will have to wait until the next active period begins. This effect decreases at increasing allocated CPU shares, since in these cases the CPU is almost dedicated to the application and whenever a request arrives it is served. The mean response time, as shown in Figure 8, seems not to be affected greatly given that the percentage of CPU assigned is the same.

In Figure 9, the comparison between the collected values of response times is shown for two different numbers of users. The difference especially in the maximum values of the

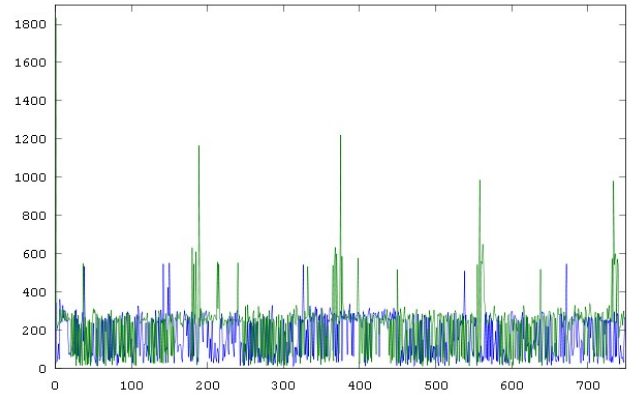

Figure 9: Comparison of various number of users (blue 30, green 50) for the same CPU share (40%) and P.

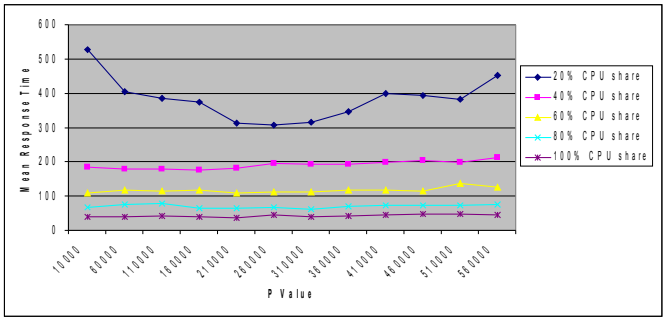

Figure 10: Different P's and CPU shares for 110 users.

distributions depicts the effect of the application workload in the response times.

In Figure 10 all the different configurations are shown for two different numbers of users. In this case, each group of columns (the first high one followed by 4 lower ones) represents one P configuration for different percentages. The upper line is for low utilization. While the utilization increases the response time decreases. In the horizontal axis the different P configurations represent increasing P values.

From these measurements it seems interesting that the best granularity (P) should depend also on the percentage of the CPU assigned to the application. In this occasion, for low percentages of utilization it is best to assign values near the middle of the investigated interval (10000-560000 us), as is depicted in Figure 10. For higher percentages of utilisation, lower values of P are more beneficial for the response times of the application. Furthermore, Figure 10 highlights the effect of the increased CPU share allocation to the response time.

C. Prediction accuracy of the ANN Model

For the estimation of the ANN accuracy, about 30% (87 test executions) of the data set was used only for validation. After the training of the model with the 70% of the test cases, we applied the according inputs of the validation cases and compared the estimated output with the observed one. The overall accuracy was around 2.5% and the error of the network for each individual test case appears in Figure 11. For each validation case, the network error appears in Figure 12.

The accuracy of the ANN models is evident from these measurements, giving sufficient reliability for this part of the overall model. The maximum deviation from the validation cases is very satisfying and so is the mean error for all the experiments. Furthermore, the predictions are not biased, a factor that is critical for the merging of different modelling approaches like in this paper.

VII. Conclusions and Future Work

In this work, we discussed how a real e-Learning distributed application has been deployed with predictable and stable QoS levels within the IRMOS platform.

We showed how we flanked the temporal isolation mechanisms available within the platform with proper

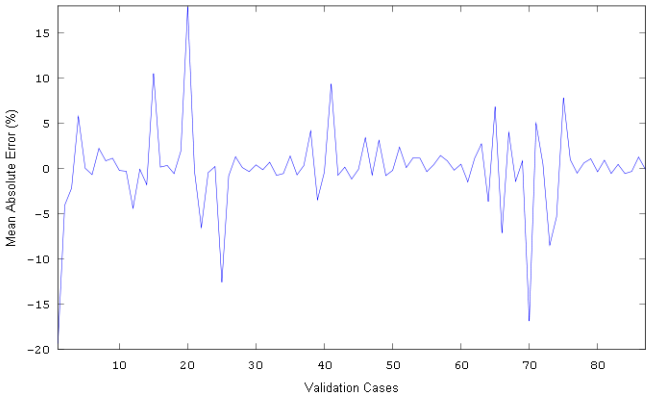

Figure 11: Accuracy of the ANN for Mean Response Time Prediction.

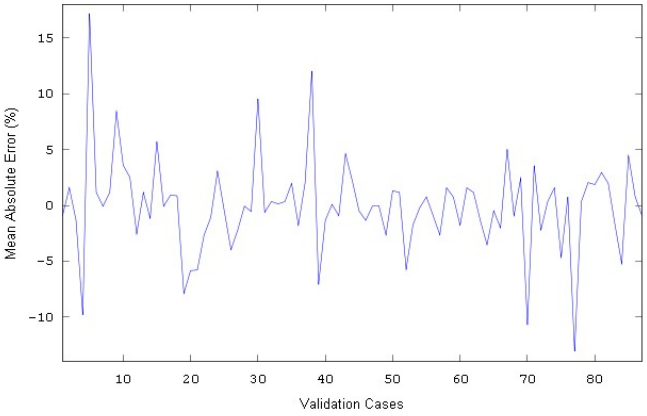

Figure 12: Accuracy of the ANN for Standard Deviation Prediction.

performance analysis, modelling and benchmarking techniques, in order to investigate the performance levels achievable by the application under the various possible configurations.

In the future, we plan to leverage the black-box approach for performance estimation, so as to apply the described technique to other applications that are already being adapted for deployment within IRMOS, such as distributed editing of professional-quality video and a virtual reality application. Also, we plan to extend the used performance models by accounting for possibly heterogeneous hardware within an ISONI domain. Finally, we plan to extensively compare the predicted performance and the actually realised one, in

presence of a variety of other deployed workload types, from compute-intensive to network-intensive ones.

References

[1] G. Kousiouris, F. Checconi, A. Mazzetti, Z. Zlatev, J. Papay, T. Voith, D. Kyriazis: Distributed Interactive Real-time Multimedia Applications: A Sampling and Analysis Framework, in 1st International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2010), Brussels, Belgium, July 2010.

[2] J.J. Moré: The Levenberg-Marquardt algorithm: implementation and theory. In: Watson, G.A. (ed.) Numerical Analysis, Dundee 1977. Lecture Notes in Mathematics, vol. 630, pp. 105–116. Springer, Berlin.

[3] B. Shirazi, L. Welch, B. Ravindran, C. Cavanaugh, B. Yanamula, R. Brucks, E. Huh: DynBench: A Dynamic Benchmark Suite for Distributed Real-Time Systems. IPDPS Workshop on Embedded HPC Systems and Applications. S. Juan, Puerto Rico, 1999.

[4] B. Lin, P. Dinda: Vsched: Mixing batch and interactive virtual machines using periodic real-time scheduling. In: Proc. of the IEEE/ACM Conf. on Supercomputing, p. 8, Nov. 2005

[5] C. Germain, C. Loomis, J.T. Mòscicki, R. Texier: Scheduling for responsive Grids. J. Grid Computing 6(1), 15–27, 2008.

[6] B. Lin, P. Dinda: Towards Scheduling Virtual Machines Based On Direct User Input. In Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing (November 2006). Virtualization Technology in Distributed Computing. IEEE Computer Society, Washington, DC, 2006.

[7] F. Checconi, T. Cucinotta, D. Faggioli, G. Lipari: Hierarchical Multiprocessor CPU Reservations for the Linux Kernel, in Proceedings of the 5th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT 2009), Dublin, Ireland, June 2009.

[8] L. Abeni and G. Buttazzo: Integrating Multimedia Applications in Hard Real-Time Systems, in Proc. IEEE Real-Time Systems Symposium, Madrid, Spain, 1998.

[9] C.L. Liu, J. W. Layland: Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. J. ACM 20, 1, pp. 46-61, Jan. 1973.

[10] M. Addis, Z. Zlatev, B. Mitchell, M. Boniface: Modelling Interactive Real-time Applications on Service Oriented Infrastructures, Proceedings of 2009 NEM Summit, ISBN 978-3-00-028953-8.

[11] Z. He, C. Peng, A. Mok: A Performance Estimation Tool for Video Applications, Proc. 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06), pp. 267-276, 2006.

[12] J.W. Lee, K. Asanovic: METERG: Measurement-Based End-to-End Performance Estimation Technique in QoS-Capable Multiprocessors, 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06), pp. 135-147, 2006.

[13] S. Benkner, G. Engelbrecht, A Generic QoS Infrastructure for Grid Web Services, Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT-ICIW'06), p. 141, 2006

[14] P. Menage: CGROUPS, October 2008. Available on-line at: http://www.mjmwired.net/kernel/Documentation/cgroups.txt.

[15] T. Voith, M. Kessler, K. Oberle, D. Lamp, A. Cuevas, P. Mandic, A. Reifert: ISONI Whitepaper, September 2008.

[16] D. Gupta, L. Cherkasova, R. Gardner, A. Vahdat: Enforcing Performance Isolation Across Virtual Machines in Xen. Proceedings of the 7th International Middleware Conference (Middleware 2006), Lecture Notes in Computer Science, Vol. 4290/2006, pp.342-362, Melbourne, Australia, November 2006.

[17] R. Nathuji, A. Kansal, A. Ghaffarkhah: Q-Clouds: Managing Performance Interference Effects for QoS-Aware Clouds, Proc. of the 5th European conference on Computer systems (EuroSys 2010), Paris, France, April 2010.

[18] L. Cherkasova, D. Gupta, A. Vahdat: Comparison of the Three CPU Schedulers in Xen, Performance Evaluation Review. Vol. 35, No.2, 30 June 2010.

[19] T. Cucinotta, G. Anastasi, L. Abeni: Respecting temporal constraints in virtualised services, Proceedings of the 2nd IEEE International Workshop on Real-Time Service-Oriented Architecture and Applications (RTSOAA 2009), Seattle, Washington, July 2009.

[20] T. Cucinotta, D. Giani, D. Faggioli, F. Checconi: Providing Performance Guarantees to Virtual Machines using Real-Time Scheduling, to appear in Proc. of the 5th Workshop on Virtualization and High-Performance Cloud Computing (VHPC 2010), Ischia (Naples), Italy, August 2010.

[21] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss: RFC2475, An Architecture for Differentiated Service. IETF, Dec. 1998.

[22] J. Wroclawski: RFC2211, Specification of the Controlled Load Quality of Service. IETF, September 1997.

[23] J. Wroclawski: RFC 2210, The Use of RSVP with IETF Integrated Services. IETF, September 1997.

[24] E. Rosen, A. Viswanathan, R. Callon: RFC3031, Multiprotocol Label Switching Architecture, IETF, January 2001.

[25] G. Kousiouris, D. Kyriazis, K. Konstanteli, S. Gogouvitis, G. Katsaros, T. A. Varvarigou, A Service-Oriented Framework for GNU Octave-Based Performance Prediction, Proceedings of the 2010 IEEE International Conference on Services Computing (SCC), pp. 114 – 121, Miami, FL, August 2010.

[26] D. Kyriazis, K. Tserpes, A. Menychtas, I. Sarantidis, T. A. Varvarigou, Service selection and workflow mapping for Grids: an approach exploiting quality-of-service information. Concurrency and Computation: Practice and Experience, Vol. 21(6): 739-766 , 2009.